AFIT/GCS/ENG/96D-04

A DOMAIN INDEPENDENT FRAMEWORK FOR DEVELOPING KNOWLEDGE BASED COMPUTER GENERATED FORCES

THESIS

James L. Benslay, Jr.
First Lieutenant, USAF

AFIT/GCS/ENG/96D-04

Approved for public release; distribution unlimited

AFIT/GCS/ENG/96D-04

A DOMAIN INDEPENDENT FRAMEWORK FOR DEVELOPING KNOWLEDGE BASED COMPUTER GENERATED FORCES

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

James L. Benslay, Jr., B.S. Computer Science

First Lieutenant, USAF

December, 1996

Approved for public release; distribution unlimited

Acknowledgments

The wise man in the storm prays to God, not for safety from danger, but for deliverance from fear. - Emerson

Only the good Lord knows the true extent of the trials faced during the last 18 months. I could not have done it without you Jesus - I praise and thank you for your steadfastness.

To my committee, I thank you for your time and patience. To Doc especially, I deeply appreciated your mentoring, and our many thought provoking engagements. I only wish that I could have been a better student for you.

To my classmates of GCS-96D, I've been blessed by our friendships. I wish you God speed wherever you may go.

To Maj Mark Kanko, God bless you sir. You have been a shelter in the storm.

To my partner in crime, Capt Vincent Brian Zurita - it's been one wild ride my friend, and I wouldn't have done it with anyone else. God bless you - (Buster).

To my best friend Travis, thanks for reminding me what our friendship was about. God bless you brother!

To my children, Josh and Sarah, please forgive me for having spent so much time away. Daddy's done now. Can we play?

To my wonderful wife, Janine, there are no words suitable to express my gratitude and love for you. You are a rare and beautiful treasure. I am forever indebted to you. My only true desire is to grow old with you and to play with our great-grand children.

Table of Contents

List of Figures

Abstract

Computer Generated Forces (CGFs) are important players in Distributed Interactive Simulation (DIS) exercises. A problem with CGFs is that they do not exhibit sufficient human behaviors to make their use effective. The SOAR approach has yielded a human cognitive model that can be applied to CGFs, but this is extremely complex. The product of the research reported in this thesis is a much less complex behavioral framework for a CGF that is easy to validate, revise, and maintain. To support this, an existing, domain independent CGF architecture is discussed and applied to an experimental CGF. Techniques for modeling the knowledge and behaviors of any CGF via semantic nets are presented. A process for transforming the semantic nets into fuzzy controllers is outlined, and pertinent issues regarding fuzzy controllers are discussed. Lastly, a method for making time critical decisions via fuzzy logic is presented.

A DOMAIN INDEPENDENT FRAMEWORK FOR DEVELOPING KNOWLEDGE BASED COMPUTER GENERATED FORCES

I. Introduction

Never tell a young person that something cannot be done. God may have been waiting for countless centuries for somebody ignorant enough of the impossibility to do that thing. - (Unknown)

1.1 Motivation

Running wargame exercises via Distributed Interactive Simulation (DIS) is a reality within the DoD. As a testamony to this, several major organizations including the Defense Modeling and Simulation Office (DMSO), the U.S. Air Force Directorate for Modeling, Simulation and Analysis, and the U.S. Army's Force XXI program, all directly promote the development of virtual wargame exercises. It has become routine for simulation centers such as the Air Force's Theater Battle Arena, and Warrior Preparation Center, as well as the Defense Advanced Research Projects Agency (DARPA) WarBreaker facility, to host highly complex simulations involving hundreds of players, both human and artificial. This virtual simulation capability has enabled the DoD to conduct scenarios that would never be possible in peacetime operations, such as simulating live fire tactical engagements by using high fidelity weapon models. Along with this opportunity, a demand has risen for better artificial players. It is currently the practice of simulation centers to augment their human players with Computer Generated Forces (CGFs). This makes the simulation more realistic by adding the necessary player components expected in a given situation, thus increasing the training value of the simulation. Indeed, this inherent ability of virtual environments to equally represent human and artificial players makes it a highly desirable tool.

One problem with the current DIS exercises, is that artificial players typically do not exhibit sufficient human behaviors to make them blend with their human player counterparts [10]. This inability to appear human in behavior gives the human adversary an advantage in deciding his course of action. By knowing which players are human and which are artificial, and having an understanding of the artificial player's responses in given situations, the human adversary has an unfair advantage and the training value is lost.

The purpose of this research is to enable more human-like behaviors in CGFs. The goal is to derive a domain independent framework that can be applied to an existing architecture whereby specific behaviors can be integrated into several levels of CGF design. An important subgoal is to keep knowledge bases highly modular so that changes to knowledge bases do not necessitate a change in system design.

1.2 Collaborative Effort

This research was conducted in close conjunction with another student's research. Capt Vincent B. Zurita's thesis *An Architecture for Computer Generated Forces in Complex Distributed Virtual Environments* [40] should be considered a companion volume to this thesis, as the concepts within each are interdependent. In addition, due to the fundamental and jointly owned concepts involved, Chapter III has been co-authored and included in both theses for clarity and completeness. For a short overview of the background material necessary to understand this thesis, refer to Appendix A.

1.3 Intelligent Wingman

The Intelligent Wingman is an experimental CGF sponsored by ESC and assembled at AFIT's Virtual Environments Laboratory. A majority of this research was conducted via the Intelligent Wingman, and while it is not expressly presented in this thesis, many examples from the Intelligent Wingman are used throughout the text to illustrate the concepts presented.

1.4 Overview

1. Chapter II begins the heart of this research by exploring why the use of fuzzy logic is appropriate to this problem and how it has been applied before.
2. Chapter III considers the problem of utilizing a domain independent architecture for CGF construction. The roles of the essential subcomponents are defined that become the foundation for combining software and knowledge engineering goals. The architecture is applied to the Intelligent Wingman, and as a result, a concurrent processing perspective of the architecture is produced.
3. Chapter IV organizes generic CGF knowledge classifications, and the domains CGFs operate within. A graphical nomenclature is established for representing the knowledge relationships within the domains via a semantic net, and several examples are given.
4. Chapter V outlines how the semantic nets are transformed into fuzzy controller processes, and discusses issues related to this process.
5. Chapter VI discusses issues involving using fuzzy controllers in a comparative decision process.

II. Role of Fuzzy Logic

There is one thing stronger than all the armies in the world: and that is an idea whose time has come. - Victor Hugo

This chapter presents the purpose of using fuzzy logic and how it applies, examines previous work in this area, and describes the fuzzification process. Lastly, methods for qualifying success are discussed.

2.1 Why Use Fuzzy Logic? How Does it Apply?

Fuzzy logic is not an imprecise logic, it is a logic of imprecision. It is a way to represent the inherent inexactness that pervades our world. When we think about describing or classifying a particular thing, it is often the case that a relative qualifier is applied rather than a discrete qualifier. It is this ability of fuzzy logic to portray the inexactness of something that makes it an excellent vehicle to describe bodies of knowledge and decisions that are not absolute.

Such is the nature of human decision making. When a person makes a decision, it is done in a relativistic way that includes perspectives with varying degrees of certainty. Even seemingly concrete decisions include opposing perspectives that simply have a high degree of non-applicability. In contrast, a standard boolean logic approach to decision making requires that all perspectives fall within discrete categories of certainty: yes or no, true or false, black or white, etc.

If the goal of a CGF is to simulate the behavior of a human, why design it to think in discrete, precise terms so that its actions have discrete, predictable results? To better simulate the behavior of a human, a CGF needs the ability to think in gradients, and to have the ability to weigh sensory inputs according to their relative importance rather than assigning a singular meaning to the input. For example, consider an F-16 CGF that is poised to encounter a bandit aircraft that is 17 miles away. The F-16 is armed with a missile that has a maximum range of 15 miles. As the F-16 decreases the distance between the two aircraft, the bandit's range state changes discretely when the F-16 comes within 15 miles of the bandit. If the F-16 was using this range as the qualifier for whether or not to fire, it would do so even though perhaps the missile has a low probability of successfully reaching the target before running out of fuel. If, on the other hand, the range is modeled as a set of two fuzzy terms depicting the degree to which the bandit was in and out of range, the F-16 CGF could decide on when to shoot as a function of the degree to which the Bandit was in range, thereby increasing the probability of a successful kill.

Conversely, an argument can be made that instead of having only two discrete range states, that three or more can be employed to better describe the bandit's range. This is certainly possible, but a trade-off occurs in that the complexity of the underlying rule based system grows in complexity as more states are added to the system. By utilizing a fuzzy range state, the complexity of the system can be kept low while allowing a high degree of expression in the state spaces.

Fuzzy logic can therefore be accepted as an appropriate means for decision making, but what about the implementation of those decisions? A CGF is still a digitally controlled entity, and even though it may make a decision based on fuzzy factors, unless otherwise programmed, it will still implement those decisions with the precision of a digital entity. What is needed is to introduce a certain amount of *uncertainty* into the actions of the CGF. When humans perform actions, they seldom do so with exacting precision, especially in highly stressful situations such as combat. If a CGF is to simulate the imprecision in human behavior, then imprecision must be introduced in the CGF. Fuzzy logic can introduce this imprecision in completely scalable ways by implementating decisions as a function of any number of fuzzily defined CGF capability attributes. For instance, it goes without saying that each person is different. We all have different strengths and weaknesses which enable us to do things to varying degrees. Why not enable a CGF with the same basic degree of varying qualities? As will be shown later, it is possible to define a grouping of qualities that make up an individual, and by means of fuzzy inferencing, determine the varying degrees to which the qualities affect the implementation of decisions. Indeed, this could be a highly dynamic process wherein crucial environmental factors affect the performance of the CGF. This would allow nearly an infinite number of instantiations of a single CGF by varying the quality attributes.

Still the question can be asked, hasn't this already been done? Doesn't a model already exist? As discussed in Chapter A, there are two fundamental types of virtual entities: cognitive process models and behavioral models. An extensive amount of work has been done on the SOAR cognitive process model [18]. This is a highly complex model that's been generalized to do multi-domain problem solving [23]. A goal of this research is to obtain the same general results in behavior as SOAR without the overhead of modeling the cognitive process. This would provide a less costly and less complicated, but fully functional CGF. To date, this has yet to be done.

The originator of fuzzy logic, Lotfi A. Zadeh, summed this up well when he said,

> "What is gained through fuzzification is greater generality, higher expressive power, and enhanced ability to model real-world phenomena and, most importantly, a methodology for exploiting the tolerance for imprecision - a methodology which serves to achieve tractability, robustness and lower solution cost." [39]

2.2 Previous Work in Applying Fuzzy Logic to CGFs

This work is a direct follow-on to a proof of concept given by Edwards [9]. Other work includes Parsons [29], whose work was directed towards using the intuitiveness and efficiency in fuzzy logic controllers to simulate command and control decisions. He shows how the controller is the heart of a value driven decision making process. Work similar to this will be shown later on in Chapter VI. Stytz and Block [26] presented an application to help users pay attention to places where items of interest were occurring in large scale virtual environments. The application used a fuzzy logic controller to decide what was important enough to be paid attention to.

George and Cardullo [12] have written a paper that proposes a unified model for simulator sensor fusion that could be useful to a fuzzy CGF. Ruspini [32] has published a paper in which he uses a fuzzy logic approach to solve control problems of complex systems operating in uncertain environments. Fundamentally he asserts that fuzzy logic is the necessary approach to solve this class of problems.

2.3 What Should Be Fuzzified?

Naturally this question needs to be asked. The answer lies in the application, and to what degree the CGF's behaviors need to be modified to appear human. As will be seen later in Chapters IV and V, the process of knowledge engineering a CGF will include identifying those measurements, decisions, and actions that need to be fuzzified.

2.4 How To Approach the Fuzzification Process

Given that the state of CGF development is still very young, an incremental build and test approach should be used when engineering fuzzy controller processes. It is necessary to build and prove the foundational controllers before moving on to more advanced controllers. Chapters IV through B present a perspective on this process.

2.5 The Measure of Success

How and when will we consider this approach to building CGFs a success? Petty [30] proposes applying the Turing Test to CGFs as a direct measure of their success. Petty discusses the applicability and usefulness of the Turing Test, and what is implied by a CGF passing the test. His conclusions were that the CGF Turing Test was technically irrelevant for training situations, but that the test still provided a useful heuristic in determining the quality of the CGF system.

III. General CGF Architecture

There is no expedient to which a man will not go to avoid the real labor of thinking. - Thomas A. Edison

In order to effectively seek the solution of a problem, one must first thoroughly understand the problem. The purpose of this chapter is to explore the domain of the CGF architecture problem, examine a currently existing architecture, and to present a concurrent processing perspective of this architecture. This will be done by breaking the architecture down into its subcomponents, and further defining the functions of the essential subcomponents via semantic Areas of Concern. The concurrent processing perspective is useful for understanding the dynamics of the CGF architecture.

3.1 Define the Problem Domain

When considering the virtual environment, it is certainly plausible to envision a CGF filling the role of any of the entities within that environment. Regardless of the role of the entity, given that we have sufficient understanding of the entity's knowledge and its behavior in a specified environment, we should be able to create a CGF to fill that entity's role. If we assume for the moment that we do have an understanding of an entity's knowledge base and its intrinsic behaviors, the question then arises "How do we assemble this entity"? How would we combine our understanding of the entity's knowledge and behaviors into a software architecture? But more than this, does an architecture already exist that could be used so we wouldn't have to "reinvent the wheel"? If so, can it be adapted to suit this new entity, or has it been so tightly coupled to its current implementation that it won't meet our needs?

This is the essence of the architecture problem domain. Given that a need exists to build a CGF, is there some design architecture or methodology that can be applied to take the CGF from concept to implementation, regardless of the type of CGF that needs to be implemented? This is a more complex problem than may be at first realized. Consider the elements of Figure 3.1. Any CGF we consider will have a combination of these elements, but to identify these general types certainly doesn't appear complex. The complexity comes in when all the specific instances of these elements and their resulting data and control flows are considered.

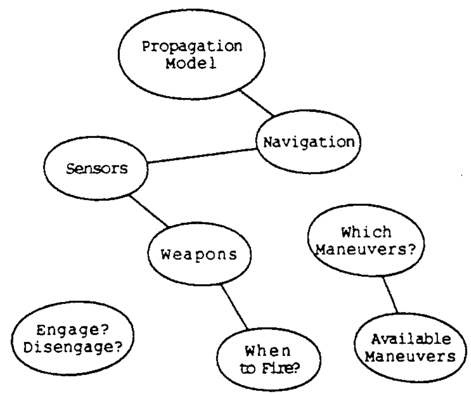

Figure 3.1 Generic elements of a CGF's domain.

Currently, it seems that whenever a simulation center implements a CGF, they do so from the ground up. What is needed is a single, domain independent solution to solve this problem.

3.2 An Existing Architecture

There is an existing architecture that takes the first step in mapping this problem domain. Santos, et al have devised a general architecture of CGF components [10]. Figure 3.2 is their model for an adaptable CGF architecture.

To summarize their work, a CGF is essentially comprised of two types of components:

1. Physical Dynamics Component (PDC)
2. Active Decisions Component (ADC)

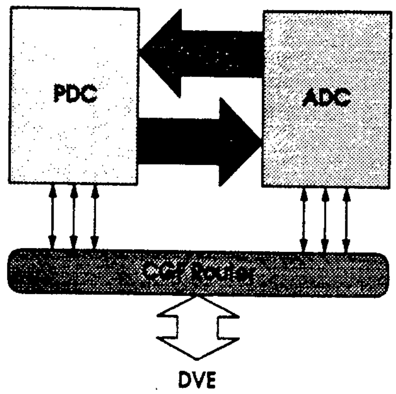

Figure 3.2 CGF architecture showing the Physical Dynamics Component, Active Decisions Component, and CGF Router.

The PDC is made up of the components necessary to model the CGF's physical makeup, such as propagation models, sensor models, weapons models, defensive elements models, etc. The PDC also contains those initialization parameters that are necessary to give the CGF a specific identity, such as performance specifications and limitations, as well as more human traits such as operator capabilities and constraints.

The ADC is composed of the components that use the information from the PDC to make decisions, and is broken down into the following subcomponents:

1. Strategic Decision Engine (SDE)
2. Tactical Decision Engine (TDE)
3. Critical Decision Engine (CDE)
4. Basic Control Module (BCM)

Each of the decision engines comprises a different level of decision making process. The SDE is concerned with high level functions such as understanding and implementing mission level goals, receiving and interpreting communications from other players, communicating with other players, and revising mission goals and subgoals.

The TDE's role is to manage the moment-to-moment operations of the entity. This entails receiving specific mission taskings from the SDE (subgoals) and implementing these according to the knowledge base of the particular entity. This will include activities such as determining what maneuvers to make in a given situation, determining when to employ ordnance, and when to implement other application specific elements such as electromagnetic counter measures.

The CDE is the survival instinct element of the virtual entity whose responsibility includes taking over control of the entity in emergency situations where termination of the entity is imminent. This is analogous to a person's instinctive reflexes transfered to the domain of the specific entity.

The BCM's role is to determine the proper propagation model inputs to perform the movements dictated by either the TDE or CDE. The BCM is considered a subcomponent of the ADC because it uses modeled behaviors to determine control inputs.

Interactions between the ADC subcomponents were defined as a number of finite state spaces and were maintained in a central communication structure.

Although not considered a component equal to the ADC or PDC, the CGF architecture has another important structure. This structure is the CGF Router (see Figure 3.2), and represents the interface between the Distributed Virtual Environment (DVE) and the two components, ADC and PDC.

The essence of this model lies in applying the concept of *separation of concerns* to the domain of CGFs. As defined in the practice of software engineering, applying a separation of concerns in a system design means separating the "how" from the "what" in the system. This is especially useful in creating an architecture for modular CGFs where we want to separate out an entity's decision making ability from its physical ability. It is also useful for scoping the entire decision making process down into manageable subcomponents. This concept of applying a separation of concerns is the foundation on which this research is based.

3.3 Essential Subcomponents

The CGF architecture was applied to the Intelligent Wingman as an integral part of this research. The idea was to build the Wingman completely around the ADC/PDC concept so that modules such as the aero model, sensor models, and behavioral components could be interchanged without disturbing the rest of the system. The first step in this process was to identify the principle subcomponents of the architecture that would be needed to begin constructing the Wingman. The following components were directly derived from the architecture model:

1. ADC Subcomponents
   (a) SDE
   (b) TDE
   (c) CDE
   (d) BCM
2. PDC Subcomponents
   (a) Propagation Model
3. CGF Router Subcomponents
   (a) DIS Interface

3.4 Semantic Areas of Concern

One of the primary purposes of Santos et al's architecture was to lay a common foundation from which a software engineer and a knowledge engineer could successfully model both the design and behavior of a CGF. Pursuant to this, the roles of the decision engines were described [10]. However, as the decision engines were being developed for the Wingman, it was necessary to more clearly define the abstracted roles of the decision engines. To accomplish this, semantic Areas Of Concern (AOC) were used. An AOC is simply a question that must be answered in order to understand the role of that decision engine. The purpose of the AOCs is two fold: first, they help the knowledge engineer identify the kinds of knowledge and behaviors that need to be modeled at the different levels of decision making, and second, they help the software engineer organize the various processes into encapsulated modules that can be replaced with like modules. In the end, the AOCs help both types of engineers make a smooth transition from the generic architecture to an applied design.

The AOCs were established as a result of modeling the real world considerations that confront a present day war fighter. We placed ourselves in the role of a "generalized" war fighter, and used our expertise as military officers to determine the knowledge that a decision engine would need in order to accomplish its generalized purpose as described by Santos et al [10]. For example, perhaps the single most fundamental question to any war fighter, regardless of service or rank, is determining one's mission. If a person or unit wants to know their function, the question is asked, "What is my mission?" Likewise, there are intuitive follow-on questions, such as "How do I go about accomplishing my mission?", and "What is my current task within the mission"? These questions are absolutely germane to both the decision engines and the role of a war fighter. In the process of enumerating the questions, they were then tailored to be domain independent, and were focused a bit more to be in line with the thinking of a CGF.

To make the AOCs as simple and direct as possible, they are written in a first person perspective from the CGF's point of view. They are designed so that by answering these questions in the context of a specific type of CGF within a specified domain, the bulk of the work for identifying the CGF's behaviors will be accomplished.

Note that these AOCs are not arranged in any intended chronological order, and that they may be tailored for a CGF type by adding new ones or by eliminating those that do not apply. The key questions are as follows:

1. SDE

    - What is my mission?
    - How do I go about accomplishing my mission?
    - Am I able to accomplish my mission?

- What is my current task within the mission?
- How do I communicate with other entities?
- How do I perceive the outside world?
- How do I deal with a lack of appropriate information?
- What are the target types that I am designed for?
- What are my responsibilities?

2. TDE

   - What kinds of maneuvers am I capable of?
   - What kinds of weapons do I know how to use?
   - How do I choose which maneuver to apply?
   - How do I know when to "take my shot"?
   - How do I know that the weapon was effective?
   - How am I aware of my environment?
   - How do I take the environment into account in deciding on a maneuver?
   - What is my tasking within this mission?

3. CDE

   - Am I in any immediate danger?
   - If I were in danger, what maneuvers do I know how to do in order to remain alive?
   - How would I decide on which maneuver to make?
   - What kinds of countermeasures do I know how to use?
   - How do I decide on when to use a countermeasure?

Defining the roles of the decision engines in this fashion reinforces two points made by Santos et al in their architecture. One, the decision engines are engaged in distinctly different levels of decision making. This distinction implies different cyclic requirements for each decision engine. Two, the decision engines rely on certain PDC components to make decisions.

A conclusion can be drawn from these two points in regards to program organization within the CGF architecture. The conclusion is that concurrent program processes should be centered around the ADC subcomponents. As for PDC subcomponents, they should also be concurrent program processes when cyclic or control requirements demand, or they can be included in the program process they are most closely tied to. An argument could be made to the contrary however, and state that each PDC subcomponent should be a separate program process and not be included in the program processes primarily devoted to ADC subcomponents. This argument goes against the opinion of established software professionals such as Gomaa [14] who state that having too many tasks increases the system overhead and complexity unnecessarily. Hence, program process consolidation should be done whenever feasible.

3.5 Concurrent Process Perspective of CGF Architecture

As a result of the above conclusion, Figure 3.3 shows the ADC subcomponents embedded within program processes of the same name. Parallelograms are used to denote program processes as established by Gomaa [14]. The program processes are depicted as interlocking ADC/PDC components to indicate the close relationship between those subcomponents within that program process. The separate graphs within the PDC portions of the program processes represent separate PDC subcomponents. Note that in keeping with the architecture design, control and state information flow are indicated by the arrows going between the interlocking components.

Because the PDC and CGF Router subcomponents identified in Section 3.3 have a high importance in the CGF architecture, they are included with the program processes of Figure 3.3 to make Figure 3.4.

Consequently, the CGF architecture can be broken down into a core of six concurrent process. These are certainly not the only program processes that could be included here, they are simply the only ones identified at this time. Identifying additional program

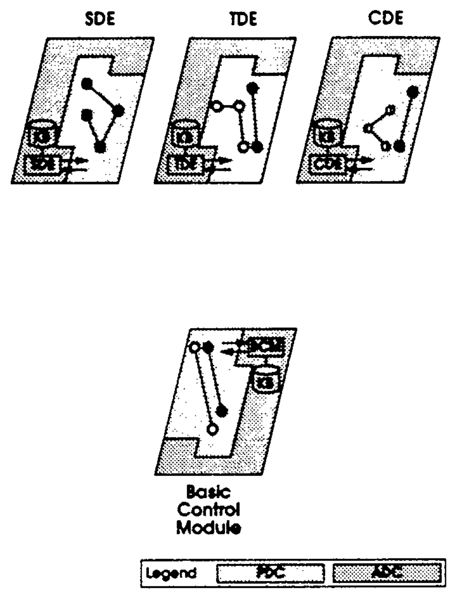

Figure 3.3 ADC and PDC subcomponents embedded within program processes.

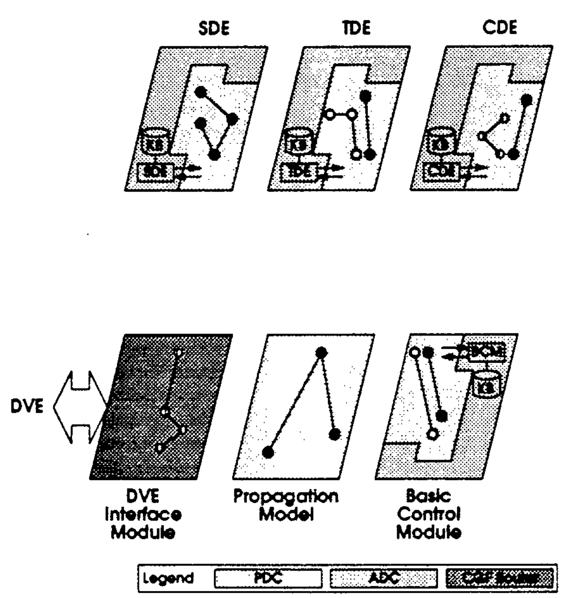

Figure 3.4 Essential PDC program processes in addition to the preceding ADC program processes.

processes would be the result of deciding a particular PDC subcomponent needed to have its own process, or perhaps determining a need for a different level of decision making.

Figure 3.4 does not show any inter-process communication occurring. In order to allow asynchronous communications between the program processes, it is necessary to expand the definition of the CGF Router to include a "Communication/Control Channel". Figure 3.5 shows how this Communication/Control Channel would interconnect all subcomponents between all the program processes. The purpose of this construct is to give any subcomponent access to the information provided by any other subcomponent, regardless of program process or cyclic requirements.

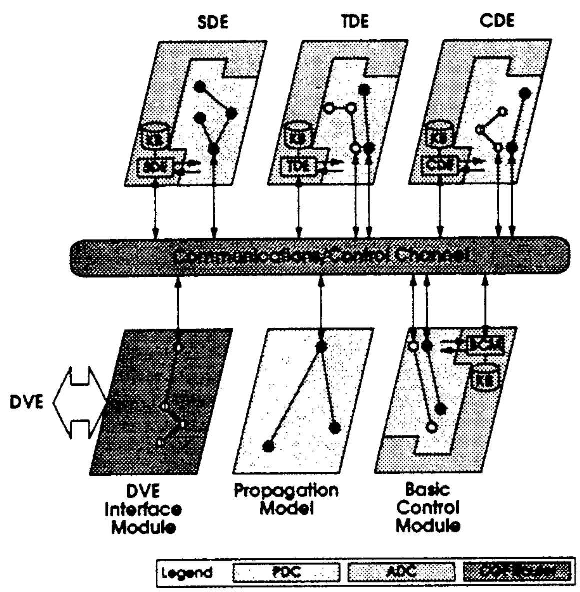

Figure 3.5 The completed concurrent process perspective of the CGF architecture.

3.6 Conclusion

By defining the decision engines according to semantic AOCs, a common foundation is established from which both software and knowledge engineers can begin the process of designing the software and behaviors of a CGF. By grouping ADC subcomponents with the PDC subcomponents they need most into the same program processes, a natural grouping of concurrent processes is established. By including the essential PDC and CGF Router subcomponents as separate program processes and combining these with the ADC program process, a concurrent process perspective of the CGF architecture is established that can be useful for understanding the dynamics of the architecture.

At this point the research separates into two paths, software engineering and knowledge engineering, that will eventually come back together. Refer to Zurita's work [40] for a software engineering perspective that takes these AOCs and categorizes them into abstract data type modules that are further refined via a series of design templates aimed at smoothing the software engineering process.

IV. CGF Knowledge and Behavior Domain Model

All knowledge has its origins in our perceptions. - Leonardo da Vinci

This chapter establishes a knowledge and behavior domain model that serves as the foundation for applying knowledge engineering to derive a class of CGFs. This is accomplished by establishing classifications of knowledge and the domains in which they belong. These classifications and domains will serve as the building blocks from which semantic nets can be constructed that both define the behavior of a CGF and form the basis of controlling processes. Issues specific to domain analysis are also discussed.

4.1 Knowledge Classifications

To enable a mapping from the model presented in this chapter to the software architecture model, it is necessary to define the types of knowledge and behaviors involved so that they can be properly placed within the context of the software architecture. For instance, it is important to differentiate between knowledge that represents how a fighter maneuver is done, from knowledge that represents an environmental measurement, from knowledge that represents a decision that changes the state of something.

The following knowledge types were actually determined *after* interviewing fighter pilot domain experts, and then trying to map the cause and effect relationships of their knowledge. Statements like "If the bandit's thus, and I'm here, then I need to do this...", or "Someone with this level of ability should be able to do this...", or "In order to determine this, then I need to know the following..." all seemed to be made up of the same basic kinds of knowledge, just arranged in different ways. For instance, a pilot's ability to detect something around him (situational awareness), and then to actually observe that thing, would require a decision be made for a reaction. These areas of knowledge could be loosely summed up as a quality (the situational awareness), a measurement (the observance), a decision, and state change (stick/throttle input). Compare this to a pilot's flying skill in completing a barrel roll. The areas could likewise be summed as a quality (the flying skill), an ability (the barrel roll), and state changes (stick/throttle input). We hypothesized that if these types of questions could be phrased generically, then they could potentially apply to any domain of war fighter. Consequently, a natural grouping of these knowledge types developed, and names were given to them to describe the generalized knowledge they represented.

1. Measurements

   Measurements are a quantitative or qualitative assessment of an environmental variable such as vehicle speed, air temperature, vehicle orientation, ammunition remaining, etc.

2. Attributes

   Attributes are parameters that describe the physical characteristics of the CGF, such as vehicle type, maximum speed, number and type of ordnance carried, radar cross-section characteristics, maximum diving depth, etc.

3. Qualities

   Qualities are those scalable characteristics that help define the behavior of the CGF. Examples are situational awareness, gunnery accuracy, experience, etc.

4. Abilities

   Abilities are those abstracted categories of skills that CGFs possess. Prime examples are basic fighter maneuvers, ground fire evasion tactics, bombing run tactics, refueling techniques, etc.

5. Decisions

   There are two types: Non-comparative and comparative. Regardless of the type, a decision always has a result in at least one term set. Changes to a term set can be viewed as a state or goal change. There is no theoretical limit to the number of states that can be changed from a single decision.

(a) Non-Comparative Decisions are those that are the result of a single inferencing process. Like a trigonometric function, input parameters are supplied and a result is turned out.

(b) Comparative Decisions are those that are the result of more than one inferencing processes that are compared together.

6. States and Goals

The idea of state and goal spaces is important to a CGF in that its very nature requires it to have a purpose and a way to identify where it is in the process of achieving that purpose. States and goals can be discrete or fuzzy, and are maintained at many levels.

4.2 CGF Domain Breakdown

Continuing with the thought of organizing the knowledge of a CGF, it is also necessary to determine the context that the knowledge types are defined in order apply meaning to them. Figure 4.1 was devised to represent these domains. The arrows depict that the domain has some affect on the other domain.

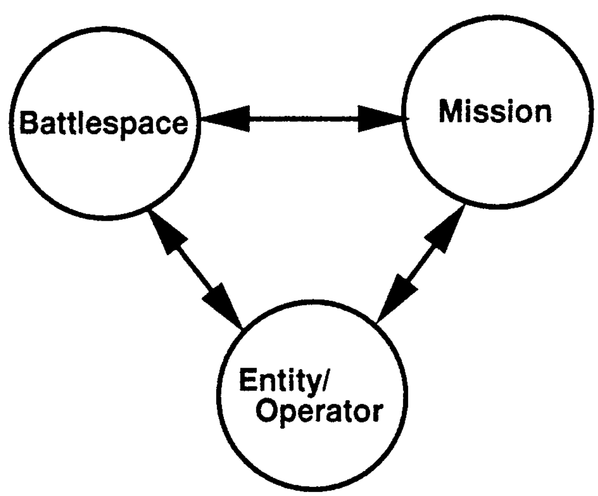

Figure 4.1 Three domains of knowledge for a CGF.

The definitions of the domains are as follows. Domains are capitalized to distinguish them from knowledge types.

1. BATTLESPACE

   Every physical thing that is exterior to the CGF is considered a part of the BATTLESPACE. Clearly this includes other CGFs, but also terrain, weather, and any other thing that can be observed and measured outside the CGF.

2. ENTITY/OPERATOR

   The ENTITY/OPERATOR is the CGF. The delineation between the two depends on the application and whether a formal definition of the operator is required. This allows a great deal of flexibility in the design process by providing scalability in how much human behavior the CGF needs to exhibit.

3. MISSION

   The abstract goals and purposes of the CGF belong in the MISSION domain. This domain is necessary to give the CGF an identity separate of its attributes.

4.3 Mapping Knowledge Types within Domains

As mentioned previously, knowledge types belong to specific domains. Figure 4.2 shows the relationship between domains and knowledge types.

The result of creating these knowledge classifications and their domains is that now the pieces exist with which a semantic net can be constructed that describes the behavior of a CGF. By identifying areas of knowledge specific to a CGF, classifying them by type, and then drawing the relationships between them, the semantics of a controlling process are created that can be mapped into a fuzzy logic controller process. In doing so, the generalized arrows in Figures 4.1 and 4.2 are replaced by specific relationships between knowledge bases. For the purposes of neatness, relationship names have been left off the arcs in the semantic nets within the following figures. In all cases, the arcs represent a "contributes to" or "determines" relationship.

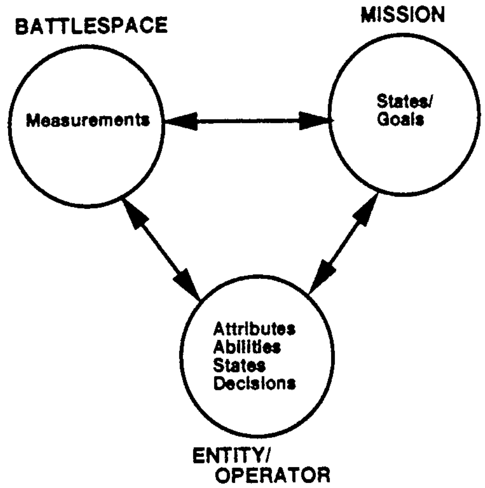

Figure 4.2 Knowledge types within their corresponding domains.

A note on domain shapes. Depending on the number of knowledge sources and relationships needed, circles may be inadequate to represent the domains. Hence, the shape of a domain is not so important as describing the boundary of its contents and where those boundaries may overlap. No restriction should be placed on what physical shape the domains take.

4.4 Mapping Knowledge Relations within Domains

It is insufficient to simply establish knowledge types within domains. What matters most is the *relationship* between the knowledge areas and how they contribute to making a decision, accomplishing a task or mission. Depending on the knowledge types used in a relationship however, the *meaning* of the relationship may change, and this must also be captured in a representation. Hence, it is necessary to establish a notation to describe the various knowledge types and their relationships within the semantic net. The goal here is to derive meaning within the net by appropriate use of symbols and relationships. Figure 4.3 shows the symbols used to represent the various knowledge types.

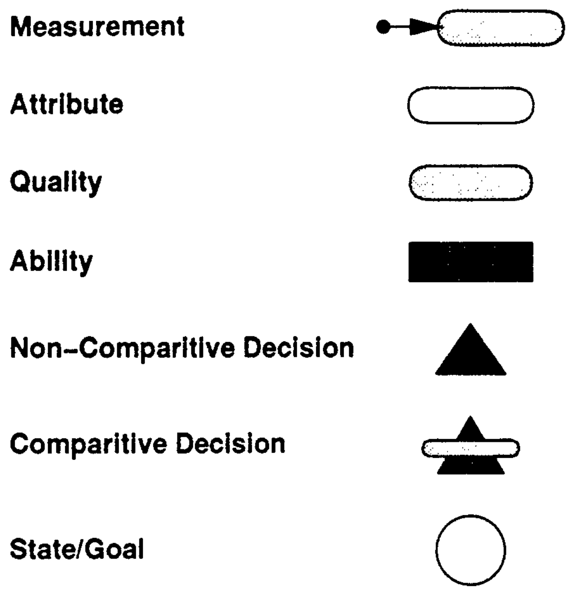

Figure 4.3 Graphical knowledge type representations.

The rest of this section is organized around presenting examples of five different combinations of knowledge types and their relationships. The first example will include the knowledge engineering process of soliciting knowledge from a domain expert and breaking that knowledge down into specific knowledge types with their respective domains and then drawing the appropriate relationships. The remaining examples will be an abbreviated form of the first example.

4.4.1 Example 1. Suppose that it is necessary to determine the mission status of an aircraft CGF while in the course of its mission. Suppose also that a knowledge engineer gains from a domain expert the following statements in regards to determining the mission status.

1. Determining mission status depends both on how attainable the mission is and the degree of the threat.
2. The degree of threat is defined by a combination of having enough situational awareness to realize a bandit exists and then knowing the relative distance to that bandit.

3. Knowing how obtainable the mission is depends on the condition of the aircraft and the abilities of the pilot.

According to these statements, the following cause and effect associations are determined:

1. knowing how attainable the mission is (and) the degree of the threat (determines) the mission status
2. having enough situational awareness (and) knowing the relative distance to bandit (determines) the degree of the threat
3. the aircraft condition (and) the pilot's abilities (determines) the mission attainability
4. the aircraft position (combined with) the bandit position (results in) the relative distance to bandit

The statements can then be distilled down to the essential knowledge elements, and propositional logic symbols used to express the knowledge combinations.

1. mission attainability $\wedge$ threat $\Longrightarrow$ mission status
2. situational awareness $\wedge$ relative distance to bandit $\Longrightarrow$ threat
3. aircraft condition $\wedge$ pilot abilities $\Longrightarrow$ mission attainability
4. aircraft position $\wedge$ bandit position $\Longrightarrow$ relative distance to bandit

Next, the knowledge engineer classifies the different pieces of knowledge and places them within their respective domains. The ϵ symbols are used to denote membership in a knowledge type or domain. Figure 4.4 shows the completed semantic net using the symbols defined in Figure 4.3.

1. mission attainability ϵ {Goals/States} ϵ {MISSION}
2. threat ϵ {Goals/States} ϵ {MISSION}
3. mission status ϵ {Goals/States} ϵ {MISSION}
4. situational awareness ϵ {Qualities} ϵ {ENTITY/OPERATOR}

5. relative distance to bandit ϵ {Measurements} ϵ {BATTLESPACE}

6. aircraft condition ϵ {Attributes} ϵ {ENTITY/OPERATOR}

7. pilot abilities ϵ {Qualities} ϵ {ENTITY/OPERATOR}

8. aircraft position ϵ {Measurements} ϵ {BATTLESPACE}

9. bandit position ϵ {Measurements} ϵ {BATTLESPACE}

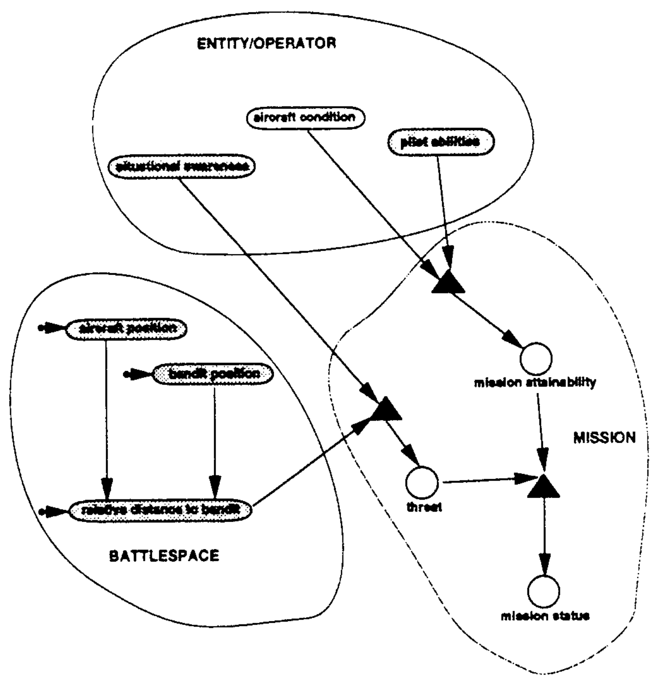

Figure 4.4 Semantic net of knowledge that describes a process of determining mission status.

This example can be generalized in the following statement: Qualities, Attributes, and Measurements determine States.

The example can also be expressed propositionally as the following:

$$Measurements \wedge Qualities \wedge Attributes \Longrightarrow Decisions$$

$$Goal \wedge Goal \Longrightarrow Decisions$$

4.4.2 Example 2. Suppose that this time the knowledge engineer is told by the domain expert that handling the pitch, roll, and yaw of an aircraft was a function of latitudinal and longitudinal pressure on the flight stick, as well as positive control of both the rudder and air brake. Furthermore, inputs to these devices are determined by how far the aircraft is off the desired point in all three planes.

Foregoing the formal classification process for this example, the generalized statement takes the form: Measurements and Measurements determine States. Propositionally, this is expressed as:

$$Measurements \wedge Measurements \Longrightarrow Decisions$$

Figure 4.5 shows the resulting relationships between the knowledge types.

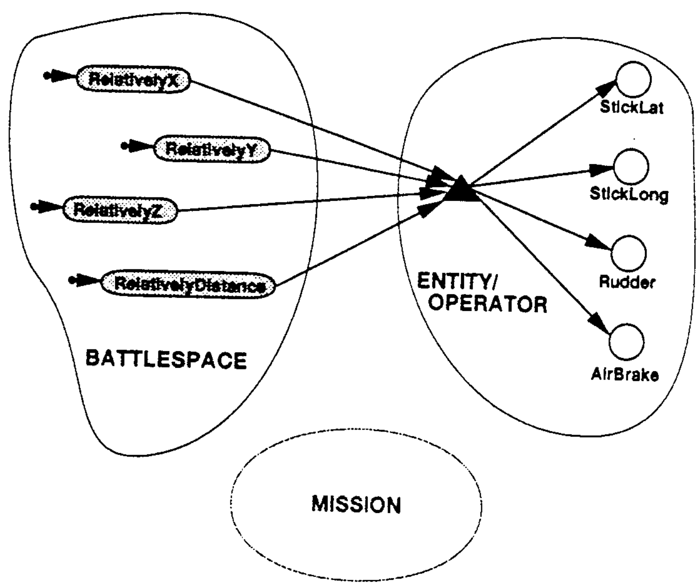

Figure 4.5 Semantic net of knowledge that describes a process for controlling aircraft orientation.

4.4.3 Example 3. Figure 4.6 shows a more complex example where the domain expert has relayed that choosing a flying maneuver to gain a position on a bandit aircraft is a combination of many factors, including but not limited to bandit aircraft types, both aircraft positions and velocities, and the types of maneuvers and weapons known for use in particular situations.

This is generalized to: Measurements and Abilities determin States. The propositional form is:

$$Measurements \wedge Abilities \Longrightarrow Decisions$$

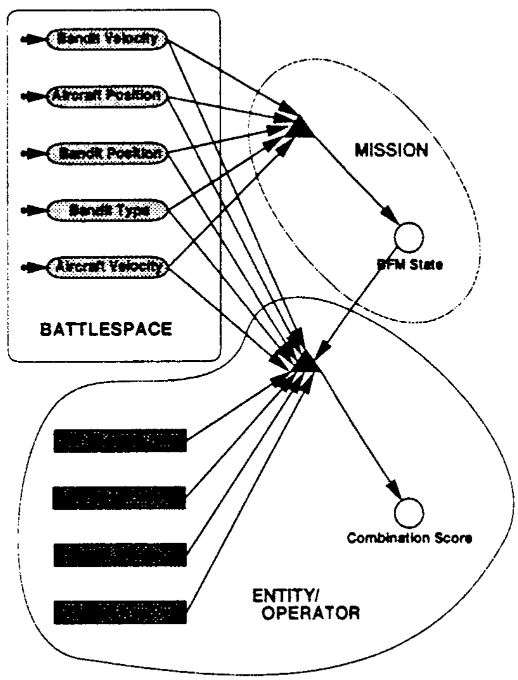

Figure 4.6 Semantic net of knowledge that describes a process for choosing a flight maneuver depending on the types of aircraft involved, their distances and velocities, as well as the known maneuvers and weapon types.

4.4.4 Example 4. Figure 4.7 reflects the domain expert's requirement that the mission goals as well as the wingman's engagement state can vary depending on the inputs from the flight lead.

This generalizes to: Measurements, States, and Goals determine States. The propositional form is:

$$Measurements \wedge States \wedge Goals \Longrightarrow Decisions$$

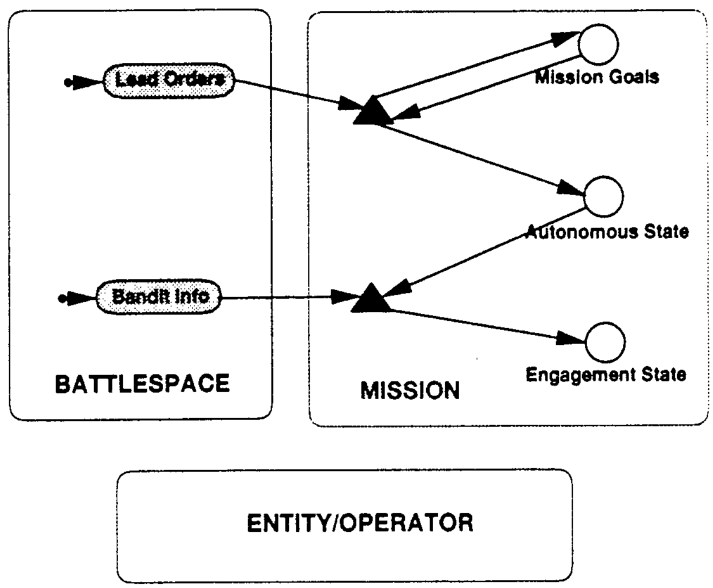

Figure 4.7 Semantic net of knowledge that describes a process for updating mission goals, autonomous state, and interception state.

4.4.5 Example 5. Figure 4.8 is a continuation of Figure 4.6. Here, the domain expert has indicated that given a sufficient amount of time, he will compare alternative maneuver options, and then choose the best maneuver possible. The semantic net becomes cluttered however, with multiple arcs going between the same two elements, so a new symbol is introduced to represent the multiple arcs. Figure 4.9 shows the new symbology. The topic of comparative decision making will be covered more indepth in Chapter VI.

Example 5 can be generalized to: Decisions and Decisions determine States. The propositional form would be:

$$Decisions \wedge Decisions \Longrightarrow Decisions$$

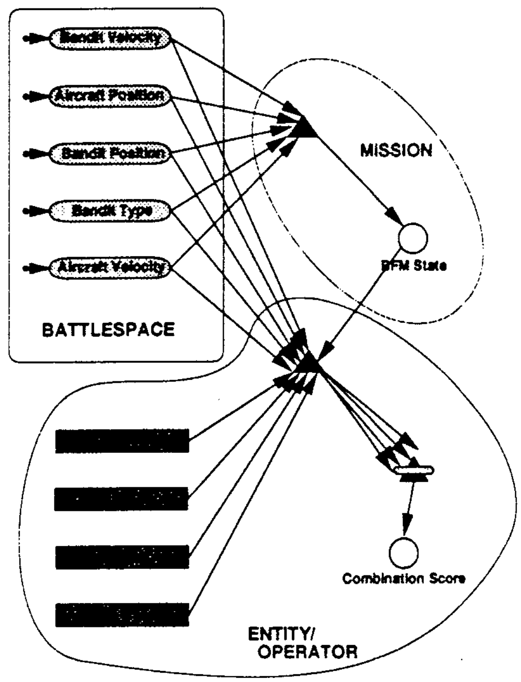

Figure 4.8 Semantic net of knowledge that describes a comparative decision process that chooses the best maneuver available.

The conclusion from these examples is that the knowledge classifications represent all the generalized types necessary to abstractly describe a CGF's behavior. Applying the types via a semnatic net within the context of the defined domains describes the relationships between knowledge, decisions, and resulting knowledge. As the behavioral requirements for the CGF become more complex however, so will identifying and defining the relationships in this semantic net. This process can be useful for keeping complexity in control by providing a framework and organization scheme to illustrate the knowledge relationship components at the lowest, meaningful level. Furthermore, defining these rela-

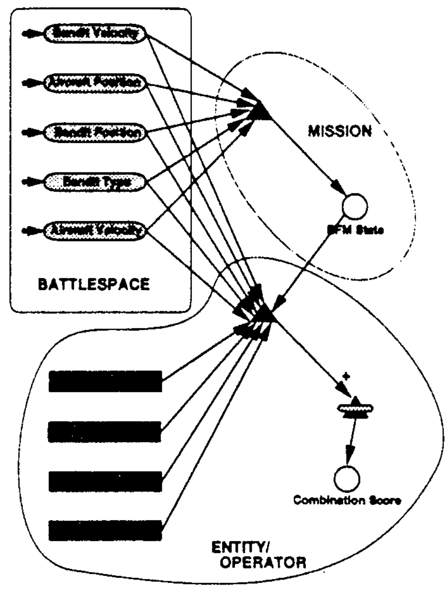

Figure 4.9 Semantic net of knowledge that an example of multiple arc symbology.

tionships is a necessary step to transforming the relationships into usable software, which is covered in Chapter V.

4.5 Specific Domain Analysis Issues

The purpose of this section is to explore some of the issues that were encountered during the course of this research. In short, there are no hard and fast answers – too much still depends on the specific implementation in question.

1. Is it necessary to employ the use of a full expert to create a CGF?

   The answer to this question lies in to what extent the CGF's behaviors need to be modeled. If the knowledge engineer possesses enough knowledge of the entity to be modeled, then he/she can serve as the domain expert. However, if the behaviors to be modeled are sufficiently complex, having the advice of a domain expert can be invaluable in obtaining a correctly behaving CGF. For the Intelligent Wingman, we used several sources as our "domain expert", including available fighter pilots, U.S.

Air Force flight manuals [34, 35], and Shaw's definitive work on fighter tactics [33]. The result was a blend of different personal insights contrasted against textbook definitions that helped us gain a breadth of knowledge on the subject, enough so that we were able to map the resulting knowledge into a semantic net and create a functional fuzzy controller to depict its behavior.

Perhaps an additional question to ask would be – is more than one domain expert necessary? As we learned from the Intelligent Wingman, involving more than one expert in the domain analysis, there is a broader base of knowledge to work from. In addition, inaccuracies in one expert's opinion may become evident as a result of another expert's input. This will only serve to bolster the accuracy of the CGF's knowledge base as inaccuracies are identified and removed.

2. How is knowledge about the CGF domain organized?

   It can seem a daunting task at first having to organize and represent the knowledge required to operate a CGF. Certainly there can be no substitute for long term exposure to a domain expert so that indepth knowledge can be gained. But what if this is not possible? How can a sufficient amount of knowledge be gained in the available time? This research has been partly directed towards this purpose. By understanding the CGF architecture model, especially the AOCs, a knowledge engineer has a head start on the knowledge elicitation process. Still, more techniques may be helpful. McEnany and Marshall [4] have reported on a detailed effort by U.S. Army's STRICOM to formalize the elicitation and organization of army unit tactical behaviors. They concentrate on Combat Instruction Sets (CISs) that enable a capture, transfer, and translation process for developing CGFs within their Close Combat Tactical Trainer.

3. What elements need to be combined to what extent to create a CGF?

   The answer to this question is similar to the first. Determining the extent of the elements necessary for a CGF is a function of the complexity of behaviors involved. By carefully reviewing and scoping the AOCs, unnecessary requirements can be eliminated early, thereby saving time and reducing complexity. As long as the domain analysis issues are properly documented, adding behaviors to the CGF is only a matter of re-initiating the domain analysis process and building on the current work.

4. How formal should a knowledge representation be?

   Depending on the application, it may be necessary to formally define by mathematical means the knowledge represented in a model. Wang and Langari [24] have proposed a formal technique for modeling complex systems where fuzzy logic is used to derive theoretical control rules of the system. If the system is a multistage decision making control model, Kacprzyk [20] has devised a formal model for describing such systems.

V. *A Fuzzy Controller Inferencing Model for the CGF Knowledge and Behavior Domain Model*

Undertake something that is difficult; it will do you good. Unless you try to do something beyond what you have already mastered, you will never grow. - Ronald E. Osborn

Up to this point, issues concerning the why of using fuzzy logic, a CGF architecture model, and a CGF knowledge domain model have been discussed. The purpose of this chapter is to map the results of the knowledge domain model into a working fuzzy logic controller. In order to do this, fuzzy controllers will be defined, and the fuzzification process broken down into sub-steps. Determining what should be fuzzy will be discussed, as well as term set models and membership function models. Reasoning Schema diagrams will be used to map behaviors into rule sets. Issues regarding stratification of rules, salience, degree of fuzziness, and defuzzification will also be discussed. Lastly, a summary of this process will be given.

5.1 Definition of a Fuzzy Controller

Klir and Yuan present a well written overview of fuzzy controllers [13]. In essence, a fuzzy controller is a specialized expert system that uses a tailored inferencing engine[1] to derive answers according to knowledge bases expressed as a set of rules. These controllers can range from simplistic ones that manage the state of a single variable, to highly complex ones that manage real time transportation systems in Japan.

Regarding the theory of fuzzy controllers, the following articles were useful in understanding the scope of fuzzy controller issues. Dubois and Prade [8] discuss how a control law can be established from a set of fuzzy rules that describes how a dynamic system reacts in various situations in order to establish behavioral control of that system. Graham and

[1]For the purposes of this chapter, the inferencing engine is generic in nature. In practice however. FuzzyCLIPS was used throughout this research.

Newell [5] discuss a method of identifying a process model from plant input-output data. They describe how the process model is represented using qualitative linguistic relationships apart from using human expertise in the design step. Sunan and Huihe [17] propose an adaptive control system that includes control mechanisms, fuzzy decisions, and neural networks. Kacprzyk [20] discusses multistage decision making and control models using fuzzy logic. He gives a series of propositions and examples that formally define multistage decision making.

Regarding generalized fuzzy expert systems theory, the scope of this research does not include considering the general application of expert systems to the domain of CGFs. For a series of papers that discuss the general topic of fuzzy expert systems theory, refer to Kandel [21].

5.2 Fuzzification Process

The process of mapping the knowledge and behavior model to a fuzzy controller can be broken down into the following steps.

1. Deciding What Should Be Fuzzy

   After considering the AOCs from Chapter III, and the knowledge modeling from Chapter IV, it is now time to decide what elements from the semantic nets need to be fuzzy. First however, it is necessary to elaborate on what knowledge types can be fuzzy. Consider again the knowledge classifications outlined in Chapter III: Measurements, Attributes, Qualities, Abilities, Decisions, States and Goals. As long as the knowledge elements in those categories can be expressed in qualitative terms, especially if the terms are poorly defined, they can be fuzzy. Essentially, this means that any piece of knowledge within the semantic net can be made to be fuzzy if desired. This is actually a bonus to the knowledge engineering process, as domain specific terms with overlapping boundaries of meaning can be applied to describe knowledge. For example, suppose that a domain expert informs a knowledge engineer that the common terms used to describe control stick movements in any direction are nudge, push, hard, and break, but that the distinctions between them are not "hard". That is to say, a hard nudge is similar to a soft push, etc. This is a prime application for describing the knowledge with fuzzy terms.

Some might say however, that Decisions can not be fuzzy. On the contrary, Dubois and Prade [8] distinguish between three different kinds of fuzzy rules, and even the FuzzyCLIPS manual allows degrees of certainty when asserting facts as a consequence of rule firings.

The question still remains, however, "What should be fuzzy"? The Intelligent Wingman took the approach that eventually all knowledge would be fuzzy, but that the fuzzification of that knowledge would be done in an evolutionary way. Even discrete areas of knowledge such as maneuver matrixes would eventually become fuzzy to some degree. The evolutionary approach would allow us time to study the effects of fuzzification of one area of knowledge before moving on to the next area.

2. Establishing Rule Sets

The semantic nets in Chapter IV use a red triangle to depict a decision making process. The process has certain inputs, and certain outputs. When mapped to the domain of fuzzy controllers, these decision making processes are in fact sets of rules in which inferencing occurs. The inputs and outputs to the decisions become antecedents and consequents to the rules. The next step is to name the rule set. Symbolically, this is done by changing the representation somewhat to assign a name. Figure 5.1 shows this change as applied to Figure 4.4. The rules themselves are not written at this time.

Note that the BATTLESPACE domain in Figure 5.1 does not include a rule set. This is because in this instance, the relative distance to the bandit is a fuzzification of the values represented by the aircraft and bandit locations. At this time, no rules are necessary in order to determine the result. Fuzzification of measurements will be discussed in detail later.

A question rises at this point as to where the rule sets should be placed within the semantic net. For instance, with regards to the rule set that determines mission attainability, an argument can be made for both cases that the rule set should reside

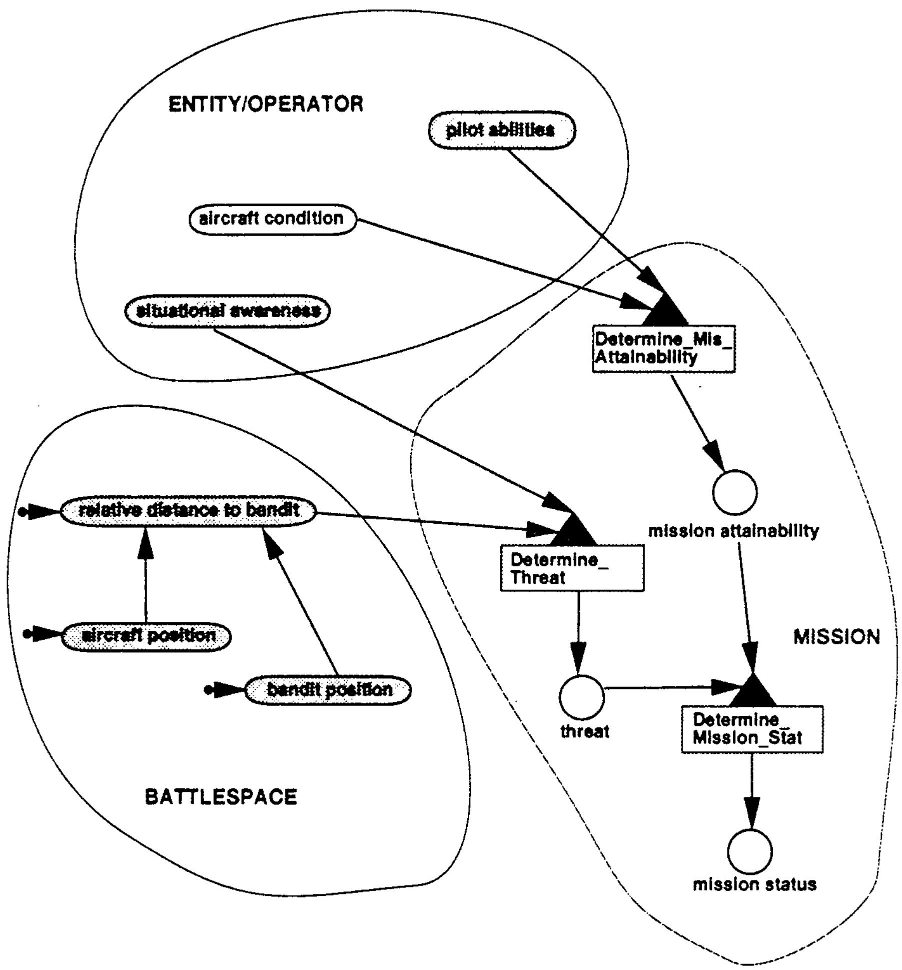

Figure 5.1 Semantic net of knowledge that includes rule set notation.

with the antecedent term sets that are considered in the rules, or that the rule set should reside the consequent term sets. As seen in Figure 4.8, rule construction can become quite complex involving multiple antecedents and multiple consequences. If both the antecedents and consequences are split among domains, where then will the rule set be placed? Considering that the purpose of a rule set is to derive some result, it seems reasonable to say that the rule set be placed in close conjunction to the term set that it affects most. In the case of a single consequence, the rule set will be in the same domain. In the case of multiple consequences across two or three domains, the rule set should be placed in the domain with the term set it seems to have the greatest impact. If this is not possible, then it is possible to represent the blended nature of the rule set by placing it in a space that is a union of the two (or three) domains. Figure 5.2 shows an example of this.

3. Term Set Model

   After rule sets have been identified within the semantic net, the next step is to establish term sets. Of the six knowledge types, all but Abilities and Decisions are transformed into term sets. This is because Abilities are only generically defined at this point to be a representation of a learned skill, such as fly a Split-S maneuver, and Decisions have already been established as rule sets.

   The process of establishing a term set is straight forward, with the bulk of the work being done when the knowledge engineer elicits the knowledge areas from the domain expert. For instance, consider the term "situational awareness" in Figure 5.1. When the domain expert is interviewed, he/she should be asked to list the terms that define the degrees of situational awareness in a rank order. With this, the term set for situational awareness is complete. Obtaining or defining the term set when the term is established is not necessary, but the term set will need to be established prior to assigning membership functions, or creating reasoning schemas for the rule sets. The term set model is represented in Figure 5.3.

   Once this is done, there is an informal notation that can be made to the semantic net to indicate the number of terms within a term set. This can be useful to convey the potential complexity of the rule sets, as the size of the rule set can suffer from

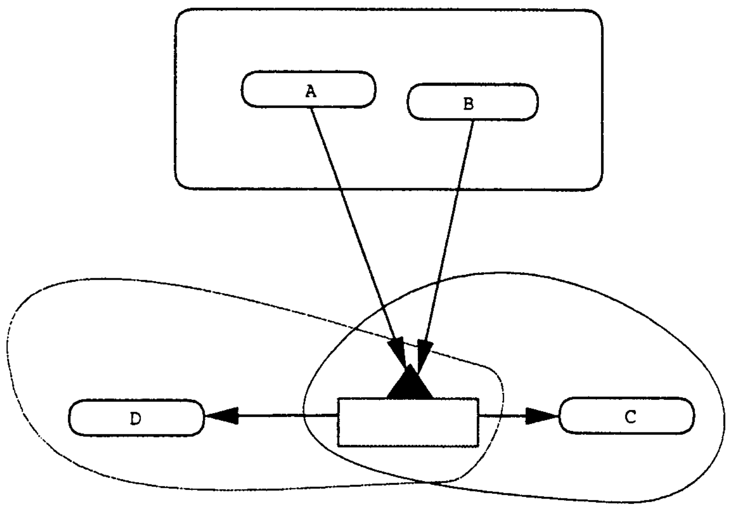

Figure 5.2 Example semantic net showing a rule set that resides in two domains.

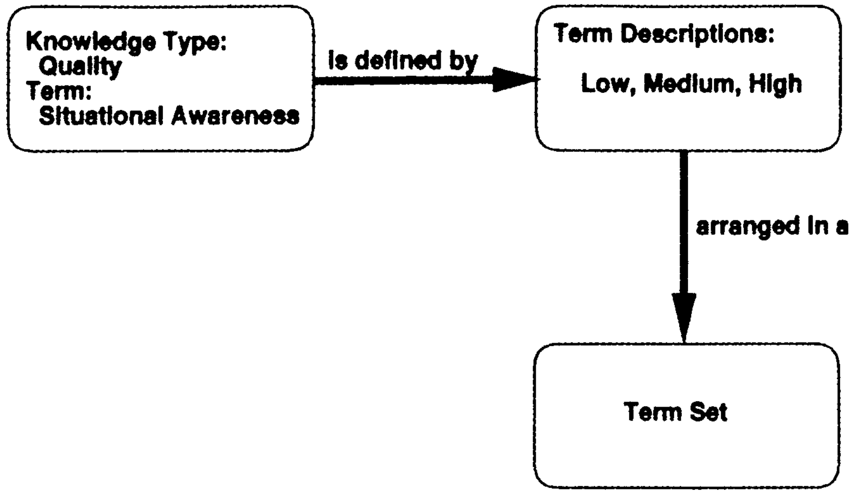

Figure 5.3 A term set model for "situational awareness".

a combinatorial explosion based on the number of antecedent terms. Ways to deal with this complexity are discussed later. Figure 5.4 shows an example of this informal notation.

4. Membership Function Model

The next step is defining the membership functions associated with each term within the term set. Two references stood out as being valuable in helping to determine the shape of a membership function. Klir and Yuan [13] have a chapter devoted to multiple techniques for membership function modeling. Turksen [36] describes a set of three basic representations of membership functions, those being the vertical, horizontal, and contour function representations. Refer to these works for an indepth coverage of this topic.

Essentially, it is the knowledge engineer's responsibility to determine from the domain expert the degree to which a particular term within a term set has membership along the range specified for the term set. When possible, empirical or statistical data should be used to define the function shape. Figure 5.5 shows the membership function model as an extension to the term set model.

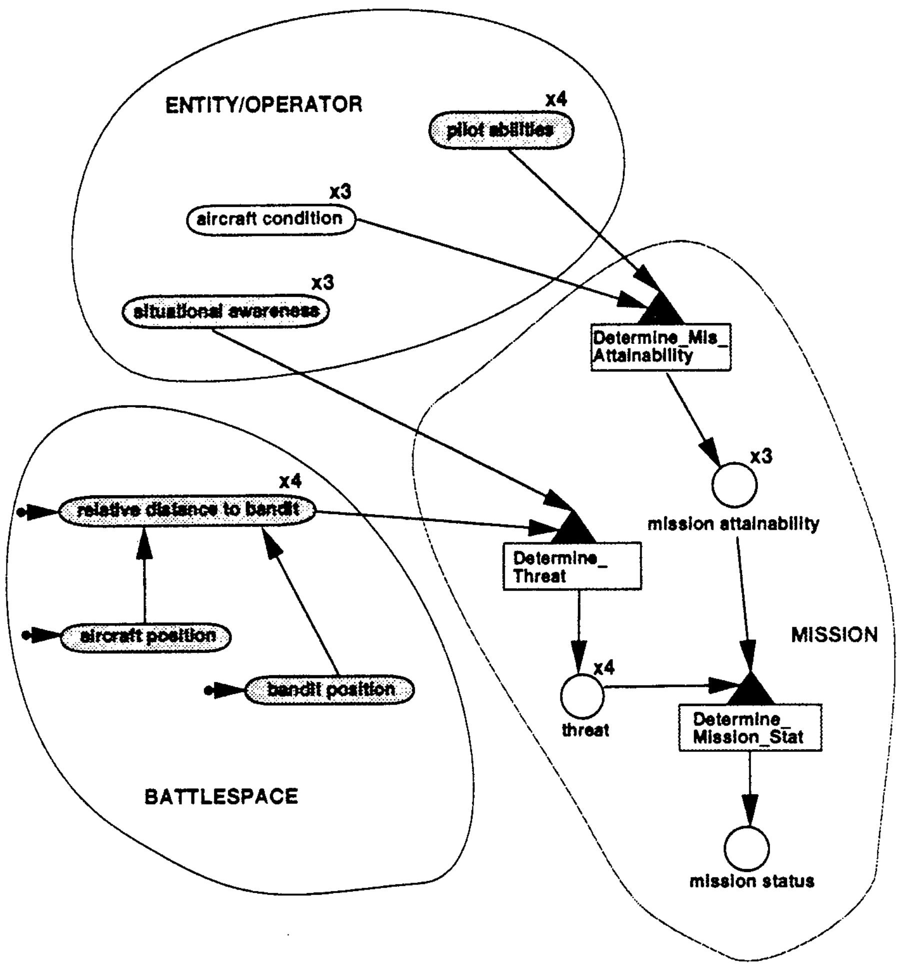

Figure 5.4 Semantic net of knowledge showing an informal notation indicating the number of terms in a term set.

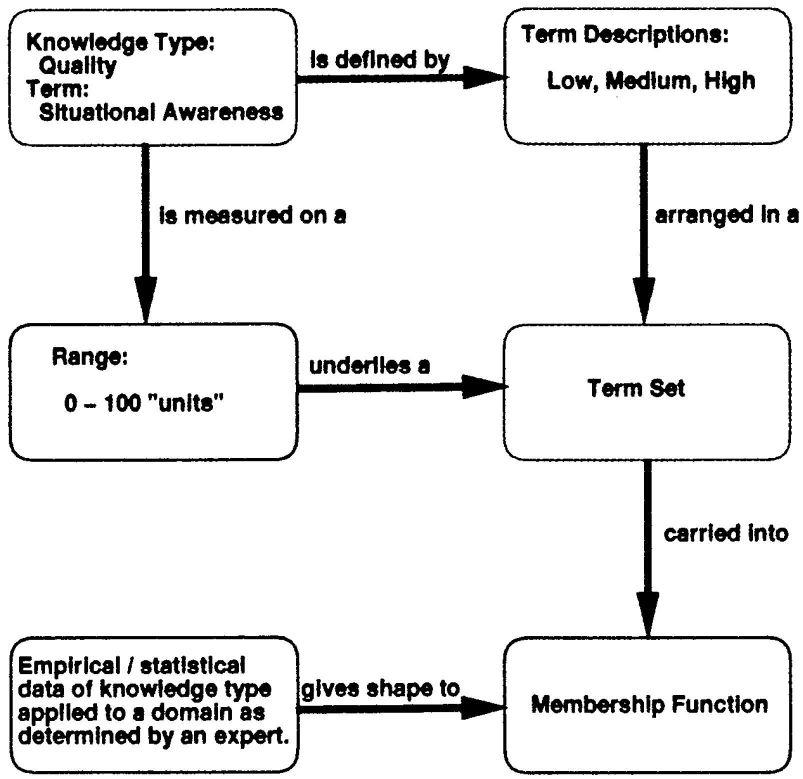

Figure 5.5 A membership function model as a continuation of the "situational awareness" term set model.

5. Rule Set Construction

It is now possible to begin constructing the rule sets. In their coverage of fuzzy controllers [13], Klir and Yuan discuss the fundamentals of charting simple inferencing rules. When more complex rules are being written, such as would be the case in Figures 4.5 and 4.6, it is necessary to use a different method. Figures 5.6 and 5.7 represent examples of *Reasoning Schemas* that were used to chart the reasoning process for the Intelligent Wingman. These Reasoning Schemas are based in part on the formalized statements by Klir and Yuan [13] that define the input variables for the inferencing engine. Refer to their work for a definition of the mathematical syntax. Note that Figures 5.6 and 5.7 only show the top portion of the schema.

Reasoning Schema
Rule: TAS
Universe Of Discourse:
Term Sets: RelativelyX × RelativelyY × RelativelyZ × RelativelyDistance ⇒ StickLat × StickLong × Rudder × AirBrake
Fact: RelativelyX × RelativelyY × RelativelyZ × RelativelyDistance is $f_e(X_o) \times f_e(Y_o) \times f_e(Z_o) \times f_e(W_o)$

Rule nomenclature:
If RelativelyX × RelativelyY × RelativelyZ × RelativelyDistance is $f_e(X_o) \times f_e(Y_o) \times f_e(Z_o) \times f_e(W_o)$,
then StickLat × StickLong × Rudder × AirBrake is:

Rule #	RelativelyX	RelativelyY	RelativelyZ	Relatively Distance	⇒	StickLat	StickLong	Rudder	AirBrake
1	behind	left_of	below	on_top_of	⇒	nudge_left	nudge_forward	nil	out
2	behind	left_of	below	near	⇒	left	nil	nil	in
3	behind	left_of	below	far_from	⇒	hard_left	nil	nil	in
4	behind	left_of	even_with	on_top_of	⇒	nil	nil	nil	in
5	behind	left_of	even_with	near	⇒	left	nil	nil	in
6	behind	left_of	even_with	far_from	⇒	hard_left	nil	nil	in
7	behind	left_of	above	on_top_of	⇒	nudge_left	nudge_back	nil	in
8	behind	left_of	above	near	⇒	left	nil	nil	in
9	behind	left_of	above	far_from	⇒	hard_left	nil	nil	in
10	behind	even_with	below	on_top_of	⇒	nil	nudge_forward	nil	in
11	behind	even_with	below	near	⇒	nil	forward	nil	in
12	behind	even_with	below	far_from	⇒	nil	hard_forward	nil	in
13	behind	even_with	even_with	on_top_of	⇒	nil	nil	nil	in
14	behind	even_with	even_with	near	⇒	nil	nil	nil	in
15	behind	even_with	even_with	far_from	⇒	nil	nil	nil	in

Figure 5.6 A Reasoning Schema diagram showing multiple antecedents and multiple consequences.

The Reasoning Schemas are the central mechanism for representing behaviors in this framework. They represent the possible changes in term sets as a consequence to the antecedent term sets being true. It is in this association of antecedent and consequent term sets that the CGF's behaviors are established. Note that the physical

Rule	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	⇒	EVALUATION
1	on_the_tail_of			hot	too_close_for	⇒	a_definite_advantage
2	on_the_tail_of			hot	within	⇒	a_pure_disadvantage
3	on_the_tail_of			hot	beyond	⇒	a_definite_disadvantage
4	on_the_tail_of			warm	too_close_for	⇒	a_definite_disadvantage
5	on_the_tail_of			warm	within	⇒	a_definite_disadvantage
6	on_the_tail_of			warm	beyond	⇒	a_definite_disadvantage
7	on_the_tail_of			cold	too_close_for	⇒	a_slight_disadvantage
8	on_the_tail_of			cold	within	⇒	a_slight_disadvantage
9	on_the_tail_of			cold	beyond	⇒	a_slight_disadvantage
10	on_the_nose_of			hot	too_close_for	⇒	no_advantage
11	on_the_nose_of			hot	within	⇒	no_advantage
12	on_the_nose_of			hot	beyond	⇒	no_advantage
13	on_the_nose_of			warm	too_close_for	⇒	a_slight_advantage
14	on_the_nose_of			warm	within	⇒	an_advantage
15	on_the_nose_of			warm	beyond	⇒	a_slight_advantage
16	on_the_nose_of			cold	too_close_for	⇒	an_advantage
17	on_the_nose_of			cold	within	⇒	a_pure_advantage
18	on_the_nose_of			cold	beyond	⇒	a_definite_advantage
19		left_of		hot	too_close_for	⇒	a_slight_disadvantage
20		left_of		hot	within	⇒	a_definite_disadvantage
21		left_of		hot	beyond	⇒	a_disadvantage
22		left_of		warm	too_close_for	⇒	a_disadvantage
23		left_of		warm	within	⇒	a_disadvantage
24		left_of		warm	beyond	⇒	a_slight_disadvantage
25		left_of		cold	too_close_for	⇒	no_advantage
26		left_of		cold	within	⇒	no_advantage

Figure 5.7 A Reasoning Schema diagram showing selective use of antecedents.

construction of the schema requires some organization. To avoid aberrant behaviors in the CGF, all possible antecedent conditions should be enumerated in the Reasoning Schema. To miss one or more conditional combinations can introduce states into the CGF's behavior that do not produce a controlling reaction when one is required.

The order that the rules occur in the schema is irrelevant – they will all be considered in the evaluation of the rule set. However, in order to make the schema easy to use and debug, the rules should be organized by placing the term set name in a column header, and placing the term set members in the column in such a way as to iterate all the combinations necessary. This may be necessary for both antecedents and consequences. Figure 5.6 is an example of this approach.

At other times, it may be useful to use a single schema to represent a grouping of rules that are similar or serve the same purpose, but that do not follow the same antecedent/consequent construction. Figure 5.7 shows an example of this approach where a blank cell in the chart means that that antecedent is not needed for that consequent combination.

An additional reason for using a Reasoning Schema is because it directly supports the concepts of scalability and expandability. This makes it a reusable tool that can last the life of the CGF. For instance, scalability is supported by two factors: rules can be incremently implemented to improve observing the system's behavior or for debugging purposes; or term sets can be set to "null" values that effectively remove their effect on the system. Figure 5.6 shows an example of this where the term set "Rudder" is always set to "nil" to discount its effect in the system. Expandability is supported by the tabular way terms are presented. If necessary, it is easy to add an additional consequent term set and identify the resulting behaviors in context to the other behaviors represented. Adding antecedents is not as easy, but still manageable by creating room within the schema where the new terms are to fit, and then filling in the required consequences.

The process of filling in the consequent portion of the Reasoning Schema is again the responsibility of the knowledge engineer. Through an interviewing process with the domain expert, the knowledge engineer determines what the correct consequent response would be to the antecedent inputs in terms of the consequent term sets. This is the essential process of modeling behaviors into the CGF. The knowledge engineer must also understand the various conjunctive and disjunctive functions that form the basis of weighted decisions in fuzzy logic. Detailed coverage of this topic can be found in several sources [1, 13, 7, 19].

There is a method to modify the interpretation of a membership function shape within the rule set via *hedges*. Hedges are adjective terms such as *very, somewhat, a little, exceedingly* that add descriptive power to a rule. The Intelligent Wingman did not use hedges in its rule constructs, but detailed discussion of their use can be found in Klir and Yuan [13], as well as Giarratano and Riley [19].

Finally, the Reasoning Schema must be transformed into a usable rule base for an inferencing engine. Note that the form and syntax of the implemented rules will vary depending on the inferencing engine used. Giarratano and Riley [19] have a comprehensive introduction to the CLIPS syntax. FuzzyCLIPS syntax can be found in the FuzzyCLIPS User's Manual [28]. As a side note, the Intelligent Wingman team found it useful to use an automated tool to take the Reasoning Schemas and produce syntactically correct FuzzyCLIPS rule sets. The tool itself was written in Ada due to the ease that Ada handles enumerated types.

6. Degree of Fuzziness Issues

   When a discrete measurement is fuzzified, it must be done so by a certain amount as comparable to the actual value of the measurement and the quantified range of the term set. Care must be taken to match these correctly. Unpredictable results that are hard to trace can result by fuzzifying a value of .0001 by 1.0. Conversely, fuzzifying a value of 1 by .0001 does little to represent the uncertainty in the data.

7. Defuzzification Issues

   If it is necessary to combine the use of an inferencing engine with other CGF code, as was the case with the Intelligent Wingman, it will be necessary at some point to *defuzzify* the information in the inferencing engine so that the exterior CGF code can use it. Although the act of defuzzifying a piece of knowledge removes from it some of its semantic meaning, algorithms have been written that attempt to arrive at the most reasonable singular quantity for a fuzzy value. The generally accepted algorithms are discussed in Klir and Yuan [13]. During the design process for the Intelligent Wingman, the Center of Gravity method and the Mean of Maxima method were compared to each other under various circumstances, and generally the Center of Gravity method gave the results that seemed the most applicable. Note that it is possible to use different defuzzification methods in different inferencing processes within the same system.

8. Stratification/Salience Issues

   Consider the semantic net in Figure 5.4. Although not discussed previously, what is happening is a stratification of rule sets. To keep the inferencing complexity low, the determination of mission status has been broken down into three separate determinations. First threat and mission attainability are determined (concurrently as their antecedents and consequences do not overlap), and then mission status is determined. Considering the fact that in the inferencing process, all rules in a rule set are evaluated to determine their applicability. This means that if there are 1000 rules in a rule set, but only 3 antecedent sets are matched, all 1000 rules will still be evaluated to determine if they are applicable. This means that the larger a rule set is, the longer the determination time. This is not generally considered a problem for a one pass inference, or when there is no time constraint on the inferencing process, but given a tight time constraint and considering the possibility that a single rule set may have to be inferenced over in the course of the same time constraint, rule set complexity then becomes an important issue.

If a stratification were not used in Figure 5.4, the mission status rule set would balloon into 144 rules, instead of the 12 rules of its current set. This may not seem like a large increase, but this is only a small example. Consider the combination of 15 term sets, each with seven terms. This results in a huge combinatorial explosion. The drawback to stratification however, is that the fuzziness of the overall knowledge base is arbitrarily reduced, hence reducing its expressiveness. For example, if Figure 5.4 did not contain any stratification, then the weighting of 14 terms would be considered in the determination of mission status. In its current configuration however, only 7 terms are considered, thus some meaning is lost.

The result is that a tradeoff occurs between degrees of fuzziness and levels of stratification. The knowledge engineer needs to weigh the issues involved, especially timing issues imposed from outside constraints, and decide on an appropriate balance between degrees of fuzziness and levels of stratification.

As a matter of implementation, it is important to understand the role of salience in rule firings. Within CLIPS, establishing a salience for a rule is the one method for forcing a rule (or group of rules) to fire within a specified sequence relative to other rules. This is necessary for stratification, as rule firings are non-deterministic within a rule stack in CLIPS. Refer to Giarratano and Riley [19] for a discussion of salience.

5.3 Summary

Here is a concise summary of the fuzzy controller construction steps presented in this chapter.

1. Consider the semantic net to be modeled as a controller. Compare and contrast fuzziness vs. stratification issues. Rework the semantic net if necessary.
2. Transform semantic net decision notation into rule set notation. Name the rule sets.
3. Decide what elements from the semantic net should be fuzzy to represent the CGF's behavior.
4. Determine the terms that make up each fuzzy term set. Annotate the semantic net if desired.
5. Create Reasoning Schemas. Illustrate the CGF's behavior within the rule set by determining consequents for each rule. Consider the use of hedges.
6. Transform the Reasoning Schemas into a rule base for use with an inferencing engine.
7. Determine which defuzzification method to apply. Write the defuzzification algorithm into the rule base and include a defuzzification rule with an appropriate salience.
8. Refer to Appendix B for a suggested method for joining the knowledge and behavior model to the software architecture model.

VI. Comparative Decision Making

Take time to deliberate, but when the time for action has arrived, stop thinking and go in. - Napoleon Bonaparte

The purpose of this chapter is to discuss the issues involved in comparative decision making.

Figure 6.1 was an example given in Chapter IV that indicated the results from more than one inferencing process were to be compared to one another and a decision made from the comparison.

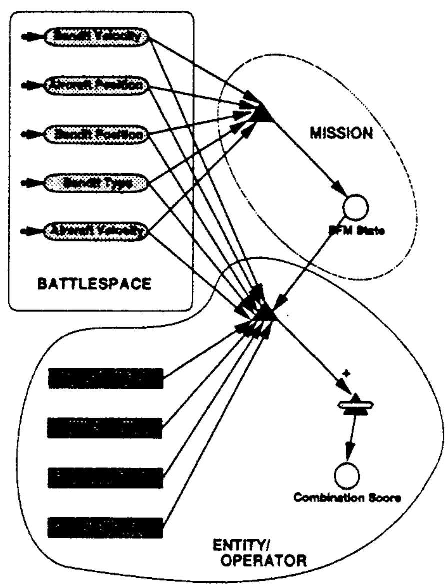

Figure 6.1 Semantic net of knowledge that shows an example of multiple arc symbology.

It should be noted that the fuzzy controller process as outlined in Chapter V is not capable of making comparative decision choices. The controller process in Chapter V is a reactive one that exhibits the behaviors designed into its rule sets. But what if the CGF needs the ability to make "what if" comparisons, and then make a decision based on the results of the "what if?" Such was the case for the Intelligent Wingman.

One of the requirements for the Intelligent Wingman was to be able to choose the "best" maneuver to employ in a real time engagement with a bandit aircraft of known abilities. This was accomplished by modifying a two player, time stepped game tree into a two player, asynchronous game tree. This algorithm analyzed the situation by determining bandit vs. wingman maneuver combinations, and then scored the resulting, relative orientations of the two aircraft according to what the algorithm expected would happen during the course of a fuzzily determined time period. The algorithm also operated within dynamically determined time constraints based on aircraft positions, velocities, as well as expected weapon ranges and velocities. The goal of the algorithm was to analyze, in a breadth first search pattern, as many maneuver combinations possible within the given time constraints, and at the end, present the best possible choice to the TDE to be implemented. The process for choosing which maneuver combination to suggest was a discrete choice and not a fuzzy one, and neither was the final representation of the chosen combination. Hence, this is another example of a decision process and a knowledge representation within the semantic net that is not fuzzy.

The problem is how to address the issue of representing this time constrained, comparative process within the context of the existing semantic net. Figure 6.2 shows how this knowledge is represented by a decision process that determines the maximum allowable time, which is then an input to the comparative decision making process. Note that the implementation of this decision making process is completely abstracted from the fact that a comparative decision is being made. In this way, the semantic net can be implemented using whatever comparative decision algorithm that meets the CGF's needs.

There were two references found during the course of the research that seemed applicable in the area of comparative decisions. The most important one was by Schaper et al [15]. Their work centered on proving the feasibility of using adversarial game trees in a real time context. The Intelligent Wingman's implementation of the game tree varies by its use of fuzzy inferenced scores based on relative orientations, and the adaptation of asynchronous play. Schaper et al also came to the conclusion that even though the begining

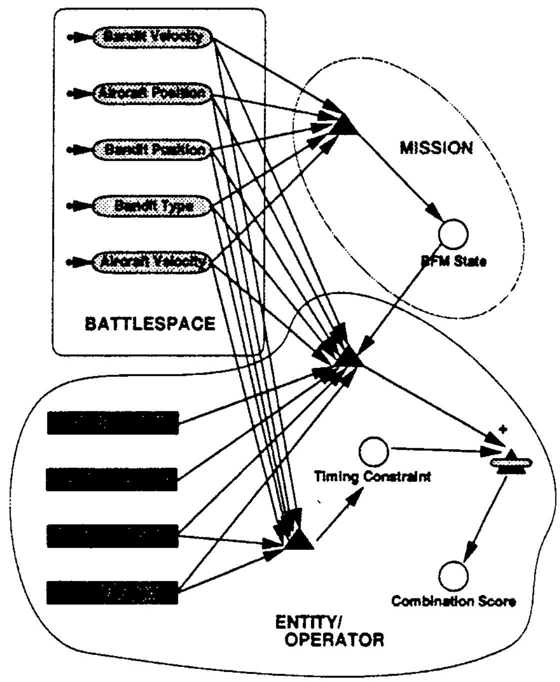

Figure 6.2 Semantic net of knowledge that includes a timing criterion for comparative decision making.

portions of a maneuver can have low scores, the maneuver can result in a high score. This same conclusion was reached during the testing of the Intelligent Wingman algorithm.

A second, albeit older reference was by Wernerfelt [37] wherein he proposed that a game scenario could be split into a set of subgames defined by sets of decisions that were made by the game players. He goes on to describe within the context of business firms, how each of the subgames could be simultaneously cooperative and non-cooperative. Whereas this does not seem appropriate within the context of a time critical decision on what maneuver to make, it could be applicable to long term decisions about what direction a mission should take.

VII. Conclusion and Future Research

Judge not of actions by their mere effect;
Dive to the center and the cause detect;
Great deeds from meanest springs
may take their course,
And smallest virtues from a mighty source.
- Alexander Pope

7.1 Conclusion

The goal of this thesis was to present a domain independent knowledge based framework wherein CGFs with human-like behaviors could be developed. Reasons for supporting the use of fuzzy logic as a means to represent real world inexactness were presented. An existing CGF architecture was presented and was adapted to a concurrent processing design. Also presented was a knowledge representation scheme to describe the intended behaviors of the CGF. Fuzzy controllers were discussed, and a process outlined by which a semantic net of CGF behaviors could be transformed into a controller process. Comparative decision making was compared against normal fuzzy controller processes. Lastly, a means was suggested for combining the results of the knowledge engineering and software engineering processes.

Still, the questions may be asked, "What was the point? Why is this framework / approach better than any other?" Aside from the obvious answers of scalability and expandability, this framework provides the capability to engineer and tune specific behaviors at multiple levels of decision making as described in Chapters IV and V. This greatly enhances an engineer's ability to validate, revise, and maintain a CGF system over the current approaches of production systems [10].

7.2 *Future Research*

There is much opportunity in this area for further study. The following are a few suggested topics.

1. A formal breakdown of human performance behaviors. This research was conducted on the advice of only a few domain experts. One of the areas that could use more thorough research is the breakdown of pilot behaviors, and how they are composed so that scalable behaviors can be programmed into the CGF.

2. There are no tools with which to build fuzzy controllers. An automated tool for assisting in the controller construction process would be a great aid.

3. Communicating between the wingman and outside agents is still problematic. In practice, there is a great deal of communication that passes between the wingman and a flight lead, but currently there is no representation in this design to show this communication, nor is there an implemention of it. Work must be done to enable commands to be passed to the CGF, and for the CGF to report information it gathers.

4. Hedges are a useful tool that have not been applied. What are the effects of applying hedges? How can they be used to refine the behaviors of rule sets?

5. Currently FuzzyCLIPS is only being used for a fraction of what it is capable of. What would be the results of moving the majority of a CGF into FuzzyCLIPS instead of the other way around? What kinds of behavioral advantages could be gained by building the decision engines entirely in FuzzyCLIPS?

Appendix A. Background

Since we cannot know all that is to be known of everything, we ought to know a little about everything. - Blaise Pascal

It is assumed that the reader has a working knowledge of the breadth of material necessary to understand this thesis. The purpose of this background section is to identify the required areas of understanding, and to point out references of essential works should the reader need further study.

References to advanced work in these topics are interspersed throughout this work as they are introduced.

1. Virtual Entities

   This work is fundamentally concerned with the interaction between entities in virtual space. These virtual entities are representative of either man in loop simulations, or CGFs (also known as Intelligent Forces or IFORS). Santos et al [10] contains introductory material to CGFs.

   CGFs can be divided into two fundamental types: cognitive and behavioral. This research is concerned with the behavioral approach to modeling CGFs. The SOAR project [38, 23, 31, 18] is an example of the cognitive approach model where the goal is to simulate the human thinking process itself and apply this process to specific problem domains. The goal of the behavioral model is to simulate human decision making without modeling the human decision making process. References to this approach are included throughout the remainder of this work.

2. Simulation Applications

   Two simulation applications were used in support of this research. ModSAF [6, 25] was used to generate virtual scenarios over a specified and topologically measured land area. AFIT's Synthetic Battle Bridge (SBB) was used to observe the virtual entity behaviors.

3. Distributed Interactive Simulation (DIS)

   DIS compatibility was a fundamental requirement from the research sponsors. Basic DIS documentation is available through the Institute for Simulation and Training [3].

4. Knowledge Representations

   Having a firm understanding of knowledge representations [2] and how to manipulate them is essential to this work. Below are three principle sub-categories that are used in various parts of this work.

   (a) Fuzzy Logic (FL)

   Fuzzy Logic is the mathematics of imprecise data. There are several established books on the subject ranging from the fundamentals [13], to arithmetic and set theory [1, 7], and applications in neural networks [22] and expert systems [21]. See Cansever and Ozguven [11] for a formal presentation of the fuzzy logic controller/inverted pendulum problem.

   (b) Rule Based Systems (RBS)

   An RBS is a collection of if-then rules which seeks to express a body of knowledge on a given subject. See Winston [38] and Hayes-Roth [16] for a comprehensive introduction.

   (c) Adversarial Game Trees

   An adversarial game tree is an approach to determining the best possible decision given a known adversary and known game rules. See Winston [38] for a basic introduction to game trees, or Morris [27] for an in depth introduction to game theory.

Appendix B. Fusion of CGF Knowledge and Behaviors with Modular Design Architecture

Creativity is essentially a lonely art. An even lonelier struggle. To some a blessing. To others a curse. It is in reality the ability to reach inside yourself and drag forth from your very soul an idea. - Lou Dorfsman

In Chapter III, a domain independent architecture for CGFs was presented, and from there the architecture was divided between the software and knowledge engineering perspectives. Chapters IV through VI dealt with the issues of breaking down and identifying the CGF knowledge and behavior domains, then how to describe semantic nets of knowledge within those domains. Fuzzy controllers were then discussed, and how to transform the CGF semantic nets into fuzzy controllers. Comparative decision making was also discussed. Now it is time to bring the knowledge engineering perspective back into contact with the software engineering perspective.

Figure B.1 shows the result of Zurita's modular CGF design [40] next to a CGF semantic net. What is required now is to identify the specific software engineering module(s) that would be responsible for encapsulating the various semantic net constructs, and then begin to physically map the relationship between the two diagrams. This would be repeated for each semantic net diagram created in the knowledge engineering process. Once accomplished, a thorough design will have been completed.

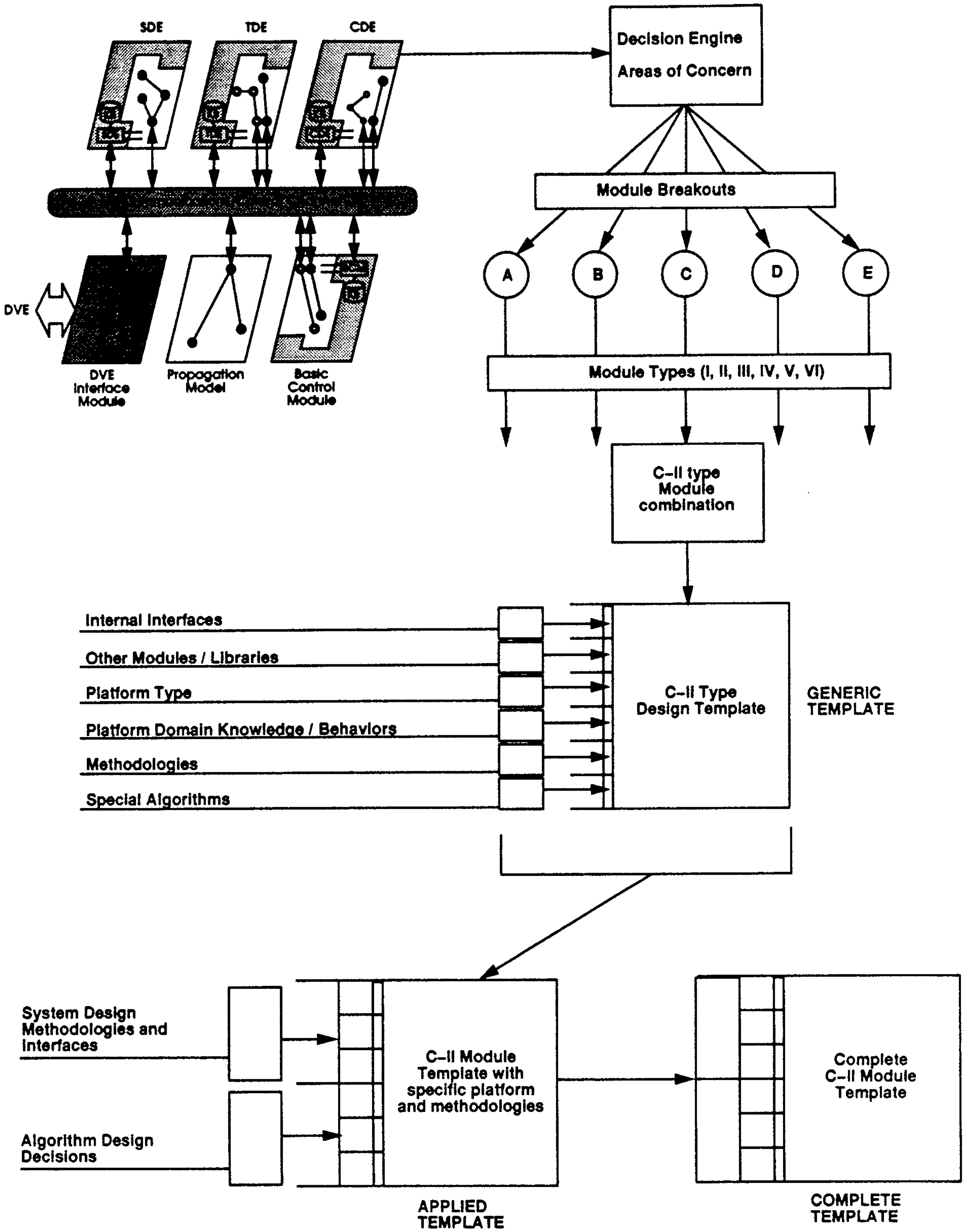

Figure B.1 Fusion of semantic net and modular CGF architecture.

Bibliography

1. Arnold Kaufmann, Madan M. Gupta. *Introduction to Fuzzy Arithmetic, Theory and Applications*. Van Nostrand Reinhold Co., 1984.

2. B. M. Kramer, J. Mylopoulos. *Encyclopedia of Artificial Intelligence*, 882–890. John Wiley and Sons, Inc., 1987.

3. Blau, B. *The DIS Protocols and their Application to Virtual Environments*. Technical Report, Institute for Simulation and Training, Orlando FL, 1994.

4. Brian R. McEnany, Henry Marshall. "CCTT SAF Functional Analysis." *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*. 195–207. May 1994.

5. Bruce Graham, Robert Newell. "An Adaptive Fuzzy Model-based Controller." *Fuzzy Logic and Fuzzy Control IJCAI '91 Workshops on Fuzzy Logic and Fuzzy Control.* Springer-Verlag, August 1991.

6. Ceranowicz, Andy. *Modular Semi-Automated Forces*. Technical Report, Loral, 1994.

7. Didier Dubois, Henri Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, 1980.

8. Didier Dubois, Henri Prade. "Basic Issues on Fuzzy Rules and Their Application to Fuzzy Control." *Fuzzy Logic and Fuzzy Control IJCAI '91 Workshops on Fuzzy Logic and Fuzzy Control.* Springer-Verlag, August 1991.

9. Edwards, Capt Mark M. *The Automated Wingman: A computer Generated Companion for Users of DIS Compatible Flight Simulators*. MS thesis, U.S. Air Force Institute of Technology, 1995.

10. Eugene Santos Jr., Sheila B. Banks, Martin R. Stytz. *Engineering Intelligent Computer Generated Forces*. Technical Report, Air Force Institute of Technology, 1996.

11. Galip Cansever, O.Faruk Ozguven. "Application of Fuzzy Set Theory to Stabilization of an Inverted Pendulum by a High Speed Fuzzy Logic Controller." *Proceedings of the 7th Meiterranean Electrotechnical Conference*. 1994.

12. Gary R. George, Frank M. Cardullo. "The Integration of Human Perception Into a Unified Mathematical Model for Developing Simulator Metrics.". (unknown proceedings title and year).

13. George J. Klir, Bo Yuan. *Fuzzy Sets and Fuzzy Logic, Theory and Applications*. Prentice-Hall Inc., 1995.

14. Gomaa, Hassan. *Software Design Methods for Concurrent and Real-Time Systems*. Addison-Wesley Publishing Company, 1993.

15. Gregory Schaper, Sridhar Pandari, Mandeep Singh. "Lookahead Limits of Intelligent Player." *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*. 401–410. IST, May 1994.

16. Hayes-Roth, F. *Encyclopedia of Artificial Intelligence*, 963–973. John Wiley and Sons, Inc., 1987.

17. Huang Sunan, Shao Huihe. "Design of Intelligent Controller." *Proceedings TENCON '93: Conference on 'Computer, Communication, Control and Power Engineering'4*, edited by Yuan Baozong. 308–11. IEEE, 1993.

18. John E. Laird, Randolph M. Jones, Paul E. Nielsen. "Coordinated Behavior of Computer Generated Forces in TacAir-Soar." *Proceedings of the Fourth Conference on Computer Generated Forces and Be4havioral Representaiton*. 325–332. May 1994.

19. Joseph Giarratano, Gary Riley. *Expert Systems, Principles and Programming*. PWS Publishing Company, 1994.

20. Kacprzyk, Janusz. "Towards 'Human Consistent' Multistage Decision Making and Control Models Using Fuzzy Sets and Fuzzy Logic," *Fuzzy Sets and Systems*, *18*:299–314 (1986).

21. Kandel, Abraham, editor. *Fuzzy Expert Systems*. CRC Press, 1992.

22. Kosko, Bart. *Neural Networks and Fuzzy Systems, A dynamical Systems Approach to Machine Intelligence*. Prentice-Hall Inc., 1992.

23. Laird, J. E. "Simulated Intelligent Forces for Air: The SOAR/IFOR Project 1995." *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL.*. May 1995.

24. Liang Wang, Reza Langari. "Complex Systems Modeling via Fuzzy Logic." *Proceedings of the 33rd IEEE Conference on Decision and Control4*. 4136–41. December 1994.

25. Loral, ADST Program Office, 12151-A Research Parkway, Orlando FL 32836. *User's Manual for ModSAF*, April 1995.

26. Martin M. Stytz, Elizabeth Block. "A Fuzzy Logic Assistant for Virtual Environment Operations Immersed in a Battlespace." *Fourth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*. 40–46. 1993.

27. Morris, Peter. *Introduction to Game Theory*. Springer-Verlag, 1994.

28. NRC. *FuzzyCLIPS Version 6.02A*, September 1994.

29. Parsons, John D. "Using Fuzzy Logic Control Technology to Simulate Human Decision Making in Warfare Models." *Proceedings of the Fourth Conference on Computer Generated Forces and Behaviroal Representation*. 519–529. DMSO, 1994.

30. Petty, Mikel D. "The Turing Test as an Evaluation Criterion for Computer Generated Forces." *Proceedings of the Fourth Conference on Computer Generated Forces and Be4havioral Representaiton*. 107–116. May 1994.

31. Randolph M. Jones, et al. "Generating Behavior in Response to Interacting Goals." *Proceedings of the Fourth Conference on Computer Generated Forces and Be4havioral Representaiton*. 317–324. May 1994.

32. Ruspini, E. H. "Fuzzy Logic-Based Planning and Reactive Control of Autonomous Mobile Robots." *Proceedings of 1995 IEEE International Conference on Fuzzy Systems3*. 1071–6. March 1995.

33. Shaw, Robert L. *Fighter Combat Tactics and Maneuvering*. Naval Institute Press, 1985.

34. TAC, PAC, USAFE. *Mission Employment Tactics, Fighter Fundamentals, F-16*, July 1991.

35. TAC, PAC, USAFE. *Combat Aircraft Fundamentals, Fighter Fundamentals, F-15E*, July 1992.

36. Turksen, I.B. "Measurement of Membership Functions and their Acquisition," *Fuzzy Sets and Systems*, *40*:5–38 (1991).

37. Wernerfelt, Birger. "Semifuzzy Games," *Fuzzy Sets and Systems*, *19*:21–28 (1986).

38. Winston, Patrick Henry. *Artificial Intelligence* (3 Edition). Addison Wesley, 1993.

39. Zadeh, Lotfi A. "Fuzzy Logic: Issues, Contentions and Perspectives." *IEEE International Conference on Acoustics, Speech and Signal Processing6*. 183. April 1994.

40. Zurita, Capt Vincent B. *An Architecture for Computer Generated Forces in Complex Distributed Virtual Environments*. MS thesis, US Air Force Institute of Technology, 1996.

Vita

James Benslay enlisted in the U.S. Air Force in March of 1987. He spent his first tour of duty as a command and control specialist at Mather AFB, CA. He was married to the former Janine Lawrence in August of 1988, and they had twins Joshua and Sarah in February of 1991. He received his B.S. Computer Sciences in 1992, graduating Magna Cum Laude. Afterwards he did a remote tour of duty in Greece, and was commissioned in the U.S. Air Force in March, 1994. His first tour as an officer was as an acquisitions manager, and worked closely with both the WarBreaker and Theater Battle Arena simulation centers on unmanned aerial vehicle programs. He was accepted to AFIT in March of 95, where he is currently finishing his master's research in applied artificial intelligence.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
December 1996

3. REPORT TYPE AND DATES COVERED
Master's Thesis

4. TITLE AND SUBTITLE
A Domain Independent Framework for Developing Knowledge Based Computer Generated Forces

5. FUNDING NUMBERS

6. AUTHOR(S)
James L. Benslay, Jr.
First Lieutenant, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/96D-04

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Electronic Systems Command
ESC/AVM
20 Schilling Circle
Hanscom AFB, MA 01731-2816

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Computer Generated Forces (CGFs) are important players in Distributed Interactive Simulation (DIS) exercises. A problem with CGFs is that they do not exhibit sufficient human behaviors to make their use effective. The SOAR approach has yielded a human cognitive model that can be applied to CGFs, but this is extremely complex. The product of the research reported in this thesis is a much less complex behavioral framework for a CGF that is easy to validate, revise, and maintain. To support this, an existing, domain independent CGF architecture is discussed and applied to an experimental CGF. Techniques for modeling the knowledge and behaviors of any CGF via semantic nets are presented. A process for transforming the semantic nets into fuzzy controllers is outlined, and pertinent issues regarding fuzzy controllers are discussed. Lastly, a method for making time critical decisions via fuzzy logic is presented.

14. SUBJECT TERMS
Distributed Interactive Simulation, Computer Generated Forces, Semi-Automated Forces (SAFOR), Distributed Virtual Environment, Fuzzy Logic, Software Architecture, Knowledged Based

15. NUMBER OF PAGES
71

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL